AND ([historical date] <= NVL([dynseg Attribute table name].End_Dte,SYSDATE).

[0198] 6) For a non-historical query of a historical Entity Attribute, add the following historical relationship to the WHERE clause, one such statement for each historical dynseg Attribute table:

AND ([Entity Attribute table name].End_Dte IS NULL).

[0199] A lookup Attribute is an Attribute that is stored as a column in a lookup table that is related to another Attribute. For example, a Paved_Ind Attribute could be implemented as a lookup Attribute associated with a Pavement_Type Attribute by adding a Paved_Ind column to the lookup table that supports the Pavement_Type Attribute. Thus, the relationship between the lookup table and parent Entity table is determined by (a) relating the lookup table to the associated Attribute and (b) relating the associated Attribute to the parent Entity table. The relationship between the lookup Attribute table and the associated Attribute table does not directly support historical queries; instead, the relationship between the associated Attribute table and the parent Entity table is used for historical queries. Note that it is allowed for the associated Attribute table to also be a lookup table. The following preferred method describes how to include a lookup Attribute table in a query, assuming the associated Attribute table is included in the query.

[0200] 1) Add the column name for the lookup Attributes to the SELECT clause.

[0201] 2) Add the table name for the lookup Attribute table to the FROM clause. The FROM clause must also include the name of the associated Attribute table.

[0202] 3) Add the following lookup Attribute relationship to the `WHERE` clause

```
AND ([associated Attribute table].[Attribute column name]
= [lookup Attribute table name].[lookup table join column
name]
```

The `WHERE` clause must also include the relationship required to support the associated Attribute table.

[0203] 4) For a historical query, no `WHERE` clause is required to directly support the lookup Attribute. Instead, whatever `WHERE` clause is required to support a historical query of the associated Attribute must be added to the SQL statement.

[0204] A relate Attribute is an Attribute that is stored as a column in a table for which there is no data-model-based relationship between the table and the parent Entity table; instead, an entry in the Relates table defines the relationship. Relate Attributes are always non-historical Attributes. The following preferred method describes how to include a relate Attribute table in a query:

[0205] 1) Add the column name of the relate Attribute to the `SELECT` clause.

[0206] 2) Add the following table names to the `FROM` clause: the name of the relate Attribute table, the name of the parent Entity table (if it is not already included in the `FROM` clause), and the names of the Relate Tables, if any.

[0207] 3) Add the `Where_Clause` from the Relates table to the `WHERE` clause.

Predefined Entity-Entity Relationships

[0208] The second type of relationship used to support queries are the Entity-Entity relationships, which define relationships between different Entity classes. For example, a Road Traversal Entity class is related to the Road Section Entity class because each road traversal is defined as a collection of sub-sections of road sections. In general, the data model defines a different type of relationship for each ordered pair of Entity classes, representing how an Entity class would be related to a parent Entity class. For example, the Traversal to Section relationship described above indicates how a Road Traversal Entity class would be related to a Road Section Entity class.

[0209] In addition to the co-located relationships defined by the data model, other relationships are supported that are defined in a Relates table. The following describes the data model based relationships, as well as how to use relationships defined in the Relates table.

[0210] The "Traversal located on Section" relationship associates each traversal with the sections of road that comprise that traversal. For example, this relationship could be used to identify all of the intersections that occur along each traversal. The following preferred method describe how to implement a Traversal located on Section relationship:

[0211] 1) Add the `[Traversal ASec table name].[traversal ID column name]` column to the `SELECT` statement.

[0212] 2) Add `the [Traversal ASec table name]` to the `FROM` statement.